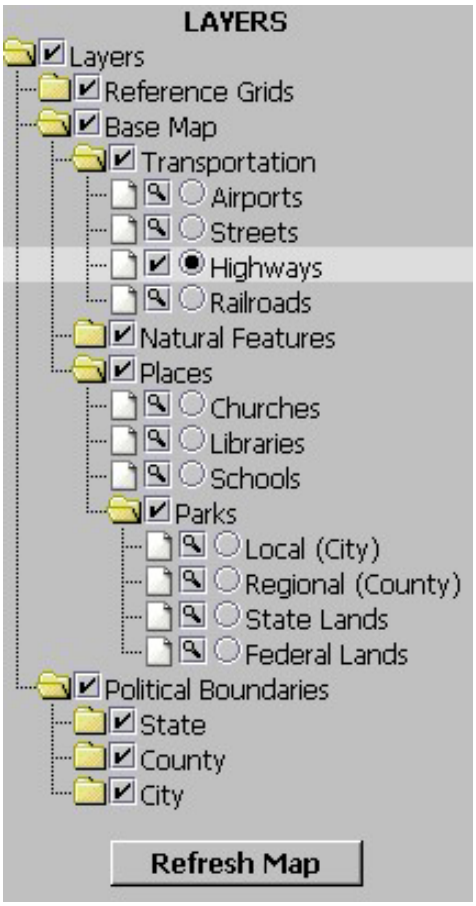
dbGroupToc Documentation

version 1.5a 02/26/2004 David Bollinger (davebollinger@hotmail.com)

Introduction



dbGroupToc is a grouping Table Of Contents (or "Layer List") for the ArcIMS HTML Viewer. It is capable of grouping any number of layers into any number of groups which may be collapsed and expanded. Groups may be nested, and layers may exist outside of a group. It supports visibility toggling at both the layer and group levels. It supports a single "swatch" per layer to indicate legend rendering, as well as an optional "legend" per layer which can include detailed legend rendering if desired. It distinguishes between layers which are completely visible, and layers which have been flagged visible but are not visible at the current zoom level. It is fairly browser-independent, requires only minimal changes to existing code to implement, and is clearly broken into code and data modules.

Considerations

This code allows for considerable flexibility in the layout of the layout list: the number of groups, the number of layers per group, the combined legend functionality, presenting groups and layers in any desired order, etc. However, this flexibility comes at a price: the legend is essentially static. It is not intended for use with a site that employs a lot of dynamic data, especially dynamic renderers. Rather it is intended for sites with a known set of data which can be carefully laid out ahead of time. It might be possible, with considerable coding effort, to make this legend behave more dynamically, but that is not its intended purpose.

This code was initially designed for the ArcIMS 3.1 HTML Viewer code set in mind, but has also been tested with the 3.0, 4.0 and 4.0.1 code sets. It has been tested with recent releases of Microsoft and Netscape browsers on the Windows platform. It is minimally functional on Netscape 4.7x era browsers on the Windows platform. It has not been thoroughly tested on older browsers. It has not been tested on other browsers, including Opera. It has not been tested on other platforms, including Mac and Unix.

Installation

1. Copy all files into a subdirectory within your web site's directory. You may place this code in any directory you wish, but these instructions and the sample data assume you will use a subdirectory named "dbGroupToc". If you decide to use a different directory name you will need to amend these instructions for your particular installation.

2. Replace the default toc.htm file, located in your web site's directory, with the toc.htm included herein. Note that these instructions assume you have not otherwise modified the default toc.htm. If your toc.htm has already been customized then you will have to amend these instructions for your particular installation.

3. Include the dbGroupToc JavaScript in MapFrame.htm either before or after the other included JavaScript files:

```
// at around line 30 of MapFrame.htm . . .
      </SCRIPT>
      <!-- dbGroupToc -->
      <SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript" SRC="dbGroupToc/dbgtCode.js"></SCRIPT>
      <SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript" SRC="dbGroupToc/dbgtMods.js"></SCRIPT>
      <SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript" SRC="dbGroupToc/dbgtData.js"></SCRIPT>
      <!-- Basic Map Display -->
      <SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript" SRC="ArcIMSparam.js"></SCRIPT>
// etc . . .
```

(note that the file dbgtMods.js is entirely optional and need not be included if not used)

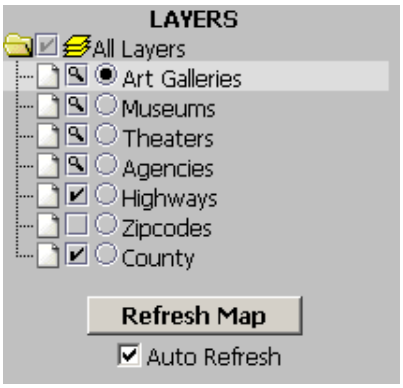
4. Customize the dbgtData.js file for your particular site. Examine the sample definitions provided and see below for further detailed instructions.

5. That's it!

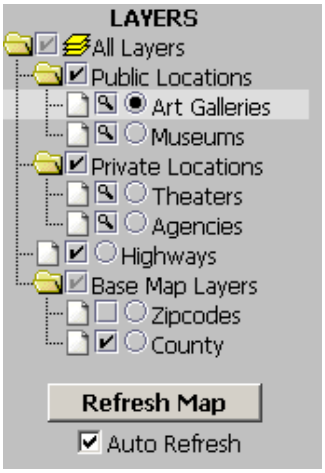
Samples / Setup

The included dbgtData.js contains a set of definitions designed to work with the “sanfrancisco” sample Map Service provided with ArcIMS. To use this sample TOC, create the Map Service according to the instructions given in $ARCIMS_HOME%\Samples\Viewers\htmlviewer\HTMLSample_setup.html for the ‘Basic Map’ sample, then use Designer to build a basic HTML Viewer site for it. Then install dbGroupToc as described above. (*Note that this legend is for demonstration only, and does not necessarily reflect the actual contents or intent of that map service.*)

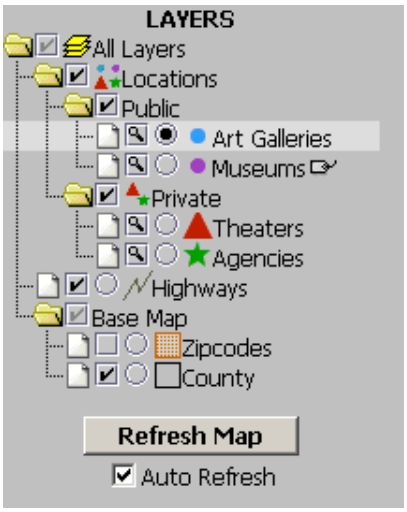
Examine the comments in dbgtCode.js - there are three basic methods for "defining" your TOC:



The first and simplest method, referred to as the "Auto Define" method, will automatically add all map layers to the TOC in a very simple non-grouped format that roughly mimics the format of ESRI's original TOC. This method does not allow you to specify swatches or legends or specific captions or any of the other custom features of this TOC. This method is the fastest way to get this TOC up and running with your map service.



The second method, referred to as the "Array Define" method, is also partially automatic but allows you to specify a single level of grouping for the layers. All map layers from the map service are automatically added and placed in groups that are specified in the array toc.LayersGroups[]. Note that the sample includes a layer (Highways) that exists outside of a group. This method does not allow you to specify swatches or legends or specific captions or any of the other custom features of this TOC. This method is a relatively quick way to add grouped layers to the TOC without having to completely define each layer individually.



The third method, referred to as the "Manual Define" method, requires you to manually specify the entire TOC with code. Any number and level of groups and layers may be freely intermixed in whatever structure you prefer. Note that the sample includes both nested groups and a layer (Highways) that exists outside of a group. This method offers full control over swatches, legends, captions and other custom features of this TOC. This method also allows you to take advantage of custom derived objects as demonstrated in dbgtMods.js. This method is the most difficult to set up for your map service, but is by far the most flexible.

Usage Notes

- This code is intended for programmers. No apologies are given for anything that I've forgotten to document! You have the code, go figure it out! :-D *(It is assumed that almost every site will desire some degree of customization.)*

- The paths and image filenames specified in the cache object must combine to form a valid URL. You may alternately leave the path variables blank and fully specify the path and filename in the filename variable. This might be useful, for example, if your swatch images were located in several different directories.

- Items are added to the toc in a top-down order. The order in which the items are displayed will match the order in which they are added. Items are also added to individual groups in a top-down order. This was done so that the code for toc definition more closely resembles the actual toc display, to ease coding. Contrast this with the typical GIS software implementation of "adding a layer" which usually implies adding it to the top of the existing list. (for example, AXL files, which read from the bottom-up)

- Not all layers in the AXL have to be added to the legend. However, any layers not added to the legend will require some other mechanism to toggle their visibility or set them active, if so desired.

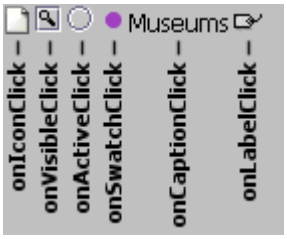
- Group visibility is implemented as a tri-state condition. If all contained layers are hidden, then the group is considered hidden. If all contained layers are visible, then the group is considered visible. If there is a combination of visible and hidden layers, then the group is considered to be in a mixed state. The default order of state change is that mixed becomes visible. This could be changed so that mixed becomes hidden by altering the value passed to setVisible() within group.writeHTML().

- You may create multiple instances of the TOC object if desired. For example, if you have some need for switching toc's "on-the-fly" it would be possible to create them all in advance, then add code to toc.htm to switch between which one of them to write out.

- To create swatches or legends I suggest you simply do a screen grab of the toc from either Author or ArcExplorer and paste the result into your favorite image editing software for further tweeking.

- Note that changes to visibility settings immediately update the internal tracking variables. What this means is that if a user changes a layer's visibility setting, even if they neglect to press the "Refresh Map" button, those visibility settings will take effect upon the next map refresh however it may be initiated. Contrast this with the behaviour of the default layer list which requires pressing the "Refresh Map" button to update layer visibility status. This behaviour is intentional.

Event Model / User Interaction



Both GROUP and LAYER objects support the same basic event model, though not all specific events may be implemented for each object, and their default handling of each particular event may differ. The illustration to the left indicates which event occurs for each of the various images and text presented within a TOC line.

Default actions for GROUP's:

onIconClick -toggle open/closed state of the group
onVisibleClick - tri-state set visiblity of the group and it's contents
onActiveClick - not currently implemented for GROUP's
onSwatchClick - no default action
onCaptionClick - toggle open/closed state of the group
onLabelClick - not currently implemented for GROUP's

Default actions for LAYER's:

onIconClick - set active layer
onVisibleClick - toggle layer visibility
onActiveClick - set active layer
onSwatchClick - toggle legend image visibility
onCaptionClick - set active layer
onLabelClick - no default action

Customization Notes

How do I change the background color of the TOC?
Look in dbgtStyle.css for the BODY element and change the background-color value. Other HTML formatting can be controlled from this file as well, such as font family, font size, margins, etc.

How do I change the color of the active layer highlight or eliminate it?
Look in dbgtStyle.css for the TR.Highlight element and change the background-color value to another color, or set it to transparent.

How do I change the folder icons and other "structural" graphics?
Look for the icon_*.gif files and replace as desired. Note that unless otherwise changed the code assumes that all such images are 16x16 pixels in size.

How do I implement multiple TOC's?
In dbgtData.js define multiple instances of the TOC object…

```
var tocOne = new TOC("TOC ONE");
// add groups and layers to tocOne
var tocTwo = new TOC("TOC TWO");
// add groups and layers to tocTwo
var toc = tocOne;  // a reference to the currently active TOC object
```

Then, to change TOC's, simply change the reference somewhere within your code:

```
// the notation "t.toc" is appropriate if this code were located in toc.htm,
// otherwise you may need to adjust the parent reference as appropriate
t.toc = tocOne; // or…
t.toc = tocTwo; // then…
t.toc.refresh();
```

(though make sure that setOutput() has been called at least once before issuing that refresh())

How do I display layer info or metadata when the layer's caption is clicked?
Add code to the LAYER.onCaptionClick() event, for example:

```
showLayerInfo(this.index);
// or
window.open('http://server/website/metadata/' + this.name + '.htm', 'InfoWindow', '');
```

How do I add a legend to a layer?
The optional legend image is specified as the fourth parameter to the LAYER constructor, simply specify a valid image filename. By default, legend images are not initially visible within the TOC, and must be toggled visible by the user clicking on the swatch image. You may alter this behavior by changing the initial value assigned to the legendVisible attribute within the LAYER constructor.

How do I implement label toggling for a layer?
This is entirely up to you, and depends on how you've set up your site. This TOC only supplies the UI portion, you need to provide the actual labelling code. The LAYER.onLabelClick() event will be the trigger, you'll need to add code there to implement it. One simple solution is to set up a separate "linked" layer for the annotation that is not added to the toc - whenever labels are turned on you set that layer visible, using the labelField variable as a place to store the id/name/index of the linked layer so that you can easily retrieve it during the event. Another solution is to define axl renderer strings in the labelField to define additional renderers for your layer, inserting those strings as necessary when building the image request. This technique will integrate well if you're using something like aimsClassRender.js.

How do I create swatch and/or legend images?
The easiest method is to just grab the legend output image from an existing website using your map service. Then use your favorite image editing software to crop out the relevant portions. A higher quality method, avoiding possible image compression present in the legend output image, is to take a screen capture of your map service within Author or ArcExplorer, and again, crop as necessary. Alternatively, if your symbology is suitable, it may be possible to just create swatches "from scratch" within your image editing software. Note that by default the code assumes that all "icon" type images will be 16x16 pixels.

I don't need to toggle labels, can I reuse the label toggle for some other purpose?
Absolutely. While you're at it, you may wish to the label icon images with something more fitting.

How do I reuse the label toggle for some other "click" function that isn't a "toggle"?
Use the same image for both the icon_labelon.gif and icon_labeloff.gif and ignore the value of "labelled". Then simply fire off the event from the onLabelClick event. An example might be to replace the icons with a "query" type image and provide quick access to a function that both sets the active layer and opens the query form.

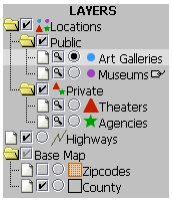
How do I create a group where only one layer can be visible at a time?
See the GROUP1 and GROUP01 classes in dbgtMods.js.

How can I merge/link separate layers for sewer lines, valve points and basin polygons (for example) into a single "Sanitation" item in the toc?
See the GROUPVL class in dbgtMods.js

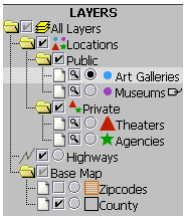
How do I make this TOC work better with Netscape 4.76 (for example, or earlier)?
This code "just barely" supports NS4 era browsers, and only as an afterthought. Such support as exists is accomplished through several "hacks" inside toc.htm -- the core code does not natively support NS4. Primarily, the code makes use of the "innerHTML" property that is not available in earlier browsers, and the hacks inside toc.htm bypass that output method and reassign the refresh() method used by the toc. It is within these hacks that you might look if you require further support for older browsers.

"Mods" - Customized Classes

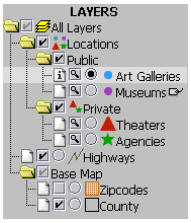
Version 1.5 introduces the dbgtMods.js module that contains a number of commonly requested modifications to the basic TOC. Use this code as a jumping off point for further customization. It demonstrates a method of pseudo-inheritance that can be used to more easily implement custom behavior within a GROUP or LAYER without having to make significant modifications to the core dbgtCode.js module. Read through the comments within dbgtMods.js to better understand what particular situation each modification is intended to address, as well as any limitations on their use.



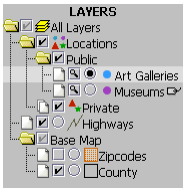
TOCNR - a TOC with **N**o **R**oot group visible - each first-level child of the root group is drawn as if it were a root node. Perhaps useful to prevent users from toggling the entire set of layers invisible, or to conserve horizontal space.



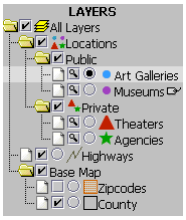
LAYERSI - a LAYER with a **S**watch **I**con - the default icon image is replaced with the layer's swatch image (see the Highways layer). Perhaps useful to conserve horizontal space or simply for stylistic preference.



LAYERAL - a LAYER with an icon indicating if it's the **A**ctive **L**ayer (see the Art Galleries layer). Best when used for all feature layers (not just one) otherwise it could be confusing. Might be used in combination with removing the active layer highlight and active layer checkbox for stylistic preference.

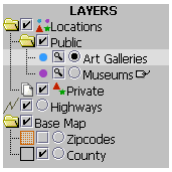


GROUPVL - a GROUP that acts like a **V**irtual **L**ayer - the group cannot be opened, and all visibility toggling affects all children of the group synchronously (see the Private group). Perhaps useful with a set of tiled image layers, or other layers that should either all be visible or hidden.



GROUP1 - a GROUP where only **1** layer may be visible at a time.
GROUP01 - a GROUP where only **0** or **1** layer may be visible at a time.

See the Base Map group. Visually these two groups appear the same, and appear the same as a normal group, but implement a radio-button-style of visibility toggling. Note the subtle difference that GROUP1 may never be completely hidden, while GROUP01 may be.



Note that you may combine mods -- once a new decendent class has been defined (and debugged!) it may be used in place of the default TOC, GROUP or LAYER objects wherever desired. For example, you may create a no-root TOCNR that contains both regular GROUP's and a GROUP1, and that GROUP1 may contain both regular LAYER's and a LAYERSI. Or any combination thereof, subject to any limitations of the decendent classes (for instance, the supplied decendent groups do not support nesting).

Class Reference

_TOC_ID_GENERATOR
About:
    For internal use only, generates unique id's for each item in the TOC.
Constructor:
    _TOC_ID_GENERATOR()
Properties:
    nextID - integer, the next value to be returned, private
Methods:
    getID() - returns the next available id, returns an integer
Instances:
    _idgen - a single global isntance

_TOC_IMAGE_CACHE
About:
    Used internally, caches frequently used images and associated helper functions.
Constructor:
    _TOC_IMAGE_CACHE()
Properties:
    numerous, image paths and images, see code for details
Methods:
    loadImage(path,file) - loads specified image file from specified path, returns Image object
    loadIcon(file) - loads specified image file from icon path, returns Image object
    loadSwatch(file) - loads specified image file from swath path, returns image object
    loadLegend(file) - loads specified image file from legend path, returns image object
Instances:
    _cache - a single global instance

TOC
About:
    The top-level TOC object.
Constructor:
    TOC(title,caption,autoRefreshMap,swatch)
        title - string
        caption - string
        autoRefreshMap - boolean
        swatch - string, filename of image to use as swatch of root group, or blank/empty string if none
Properties:
    title - string, used as title of TOC document
    caption - string, used as caption of root GROUP
    root - GROUP, the top-level group containing all other items
    isInited - boolean, indicates if TOC has been initialized, used internally
    divToc - DIV object reference, the output object for the main body of the TOC
    divTocHelp - DIV object reference, the output object for the TOC help
    autoRefreshMap = boolean, flag to automatically refresh map after TOC changes
    LayersGroups = array, contains group names for each layer, used with "Array Define" method
Events:
    onIconClick(tocid) - fires when icon is clicked, dispatched to item with specified tocid
    onVisibleClick(tocid) - fires when visible is clicked, dispatched to item with specified tocid
    onActiveClick(tocid) - fires when active is clicked, dispatched to item with specified tocid
    onSwatchClick(tocid) - fires when swatch is clicked, dispatched to item with specified tocid
    onCaptionClick(tocid) - fires when caption is clicked, dispatched to item with specified tocid

onLabelClick(tocid) - fires when label is clicked, dispatched to item with specified tocid

Methods:

init() - used internally, initializes TOC before first use

setOutput(divToc, divTocHelp) - set the output DIV objects used to write TOC contents

divToc - DIV object reference, the output object for the main body of the toc
if null then main body of the toc will not be output

divTocHelp - DIV object reference, the output object for the TOC help
if null then help will not be output

addItem(item) - adds an item to the root GROUP

item - an object reference of type GROUP or LAYER

addGroup(item) - equivalent to addItem

addLayer(item) - equivalent to addItem

refresh() - causes the TOC to redraw itself to currently specified output DIV's

refreshMap() - conditionally executes a map refresh

writeHTML() - returns a string containing the HTML output of the TOC

writeHelpHTML() - returns a string constaining the HTML output of the help

Instances:

The code as supplied expects a single global instance named "toc". Multiple instances may be created but will require additional supporting code from the developer to implement them.

GROUP

About:

Manages a collection of child GROUP and LAYER objects.

Constructor:

GROUP(caption,opened,swatch)

Properties:

parent - a reference to the object that contains this group

tocid - a unique identifier, internally maintained

caption = a string used as the caption

opened - a boolean indicating if this group is currently opened

items - an array of references to other GROUP and LAYER objects

iconOpened - an IMAGE object containing the "opened" icon image

iconClosed - an IMAGE object containing the "closed" icon image

swatch - an IMAGE object containing the swatch for this group

Events:

onIconClick() - fires when the icon image is clicked
default action is to toggle open/closed state of the group

onVisibleClick() - fires when the visible checkbox is clicked
default action is to set visiblity of the group and it's contents

onActiveClick() - fires when the active radio is clicked (not currently implemented for GROUPs)
no default action

onSwatchClick() - fires when the swatch image is clicked
no default action

onCaptionClick() - fires when the caption is clicked
default action is to toggle open/closed state of the group

onLabelClick() - fires when the label toggle image is clicked
no default action

Methods:

init() - used internally, initializes the GROUP and it's contents

findItemByTocID(tocid) - returns a reference to an object with specified tocid, or null if not found

addItem(item) - adds an item to the GROUP

item - an object reference of type GROUP or LAYER

addLayer(item) - equivalent to addItem

addGroup(item) - equivalent to addItem

getItem(index) - returns item at specified index

index - integer, zero-based index into the items array
toggleOpened() - toggles the opened/closed state of the group
setActive() - toggles the active state of the group, not currently implemented
toggleLabel() - toggles label visibility
getVisible() - returns current visible state of the group and it's contents
setVisible(value) - sets current visible state of the group and it's contents
toggleVisible() - toggles current visible state of the group and it's contents
writeHTML(child_shim, sibling_shim) - returns a string containing the HTML representation of this group in the toc.

Instances:

You may freely create as many instances as you wish, adding them to the TOC or other GROUP's as desired.

LAYER

About:

Represents a single layer in the TOC.

Constructor:

function LAYER(name,caption,swatch,legend,labelField) {

Properties:

this.parent - a reference to the object that contains the layer
this.tocid - a unique identifier, internally maintained
this.name - a string that contains the NAME or ID of this layer in the map service AXL
this.caption - a string used as the caption
this.index - an index into the layer information arrays of aimsLayers.js for the layer
this.icon - an IMAGE object containing the icon image
this.swatch - an IMAGE object containing the swatch image (if used)
this.legend - an IMAGE object containing the legend image (if used)
this.legendVisible - a boolean indicating current visible state of the legend image
this.labelField - a string for storing labelling information for the layer
this.labelled - a boolean indicating current labelled state for the layer

Events:

onIconClick() - fires when the icon image is clicked
default action is to set active layer
onVisibleClick() - fires when the visible checkbox is clicked
default action is to toggle layer visibility
onActiveClick() - fires when the active radio is clicked
default action is to set active layer
onSwatchClick() - fires when the swatch image is clicked
default action is to toggle legend image visibility
onCaptionClick() - fires when the caption is clicked
default action is to set active layer
onLabelClick() - fires when the label toggle image is clicked
no default action

Methods:

init() - initializes the layer, used internally
findItemByTocID() - returns a reference to an object with specified tocid, or null if not found
setActive() - sets the layer active
toggleLabel() - toggles labels for the layer (up to developer to implement)
getVisible() - returns current visible state of the layer
setVisible() - sets the current visible state of the layer
toggleVisible() - toggles the current visible state of the layer
writeHTML(child_shim, sibling_shim) - returns a string containing the HTML representation of this layer in the toc.

Acknowledgements

Thanks to Andrew Eddie (A.Eddie@toowoomba.qld.gov.au) for bouncing around ideas and sharing his modifications to v.1.2 (some of which are incorporated here in one way or another) and for providing me with a bit of incentive for finishing off this latest version. :-D

Thanks to all others who may have contributed ideas, suggestions, code, bug reports, etc.

History

(yanked from dbgtCode.js to reduce size)

```
// version 1.0 05/16/2002
//
// version 1.1 changes:  (only released as a patch)
//   added simple rescrolling after a refresh
//
// version 1.2 11/26/2002 changes:
//   additional testing with ArcIMS 4.0
//   object.write* methods no longer write directly to
//     a document, instead simply return an html string,
//     method names changed (postfixed with "HTML")
//   output is inteded to go to DIV's, thus fixing the
//     flicker and scrolling problems in previous version
//   toc.setOutput() method added
//   .divToc and divTocHelp added to TOC object
//   toc.refresh() method rewritten
//   easier to set optional help, simply do (or do not) supply
//     a DIV for the help when calling toc.setOutput()
//   (above changes also improve support for toc in a popup window)
//   layer info button removed, now a hyperlink around layer name
//   .iconInfo removed from RESOURCE object
//   added a sample legend image for sf demo
//   added comments inside writeHTML() methods to aid customizers
//   added these comments regarding table structure to aid customizers:
//
//     the toc table is 5 columns wide as so:
//
//     group rows:
//       | fldr | visi | groupname               |
//     layer rows:
//       | bars | visi | actv | swch | layername |
//     legend rows:
//       | emty | emty | emty | legend           |
//
//
// version 1.21 12/11/2002 changes:
//   added additional formatting in LAYER_writeHTML() to keep cells aligned
//
// version 1.3 12/11/2002 - 01/23/2003 changes: (never publicly released)
//   changes in toc.htm to support NS7
//   additional testing w/ NS7
//   title of toc is now a property of the TOC object instead of RESOURCE object
//   added autoRefreshMap property to TOC and its constructor
//   added on*Click methods as easier hooks for customization
//   removed use of displayLayerInfoButton variable
//   added alt tags with short hints for icon images
//   added nowrap attribute to group & layer name td's
//   preliminary support for nesting - add layers to toc, add groups to groups
//   minor modifications to style sheet dbgtStyle.css
//   minor code reformatting
//
// version 1.4 01/26/2003 changes:
```

```
//   major restructuring and rewrites to support nested groups/layers
//   still "mostly" backwards compatible with previous versions of data file (dbgtData.js)
//   addition of ITEM template class
//   table is now a single column with ragged alignment (no longer 5 distinct columns)
//   additional on*Click event methods added, all user interaction now passes through
//     an on*Click method, makes customization much cleaner/easier
//   added "customization note:" sections within code to (hopefully) help answer frequent questions
//   standardized use of quotes/dquotes (quotes are javascript string delims, dquotes output in html)
//   cleaned up unused attributes of dbgtResource in dbgtData.js
//   added partial support for label toggling (developer must add in actual logic)
//   more Netscape tweaking
//
// version 1.5 08/01/2003 changes:
//   toc will auto-initialize and mimic original toc if developer doesn't specify a definition
//   toc can be specified via the "Array Define" method using the LayersGroups array
//   cache images
//   moved old "resource" data from dbgtData.js into dbgtCode.js (since it was rarely
//     modified) and renamed/reused it as the image caching mechanism
//   flattened object model and distributed event handling better
//   ITEM class removed (and use of 'instanceof' removed to better support sub-classing)
//   prototype syntax used
//   TOC's event handlers no longer contain logic, just dispatching to GROUPs/LAYERs
//     (again, to better support sub-classing)
//   onFolderClick and onLayerClick event handlers combined into onIconClick
//     (since it was really the same event, just on different objects. again for sub-classing)
//   LAYER's lookup of index property now works with both id and name from mapservice
//   caption property added, to distinguish from name property used to lookup index
//     for example, lookup layer by name 'SDE.DBO.PARCELS' but caption as 'Parcels'
//   added checkbox to toggle autoRefreshMap
//   added underscores to classes/objects/functions/variables intended to be private
//   converted unique id generator into a class
//   default action of LAYER's swatch click is to hide/show legend (if provided)
//   removed unused 'value' parameter from GROUP's setVisible method
//   added swatch support for groups
//   added icon references to GROUP and LAYER objects (so could be customized)
//   added dbgtMods.js to demonstrate frequently requested modifications via subclassing
//   "customization notes" were removed from dbgtCode.js to reduce code size
//     (still lots of various comments in dbgtMods.js and dbgtData.js - it is expected
//      that the developer would want to trim these out before deployment to reduce code size,
//      but hopefully they won't have to mess with dbgtCode.js)
//
// version 1.5a (02/26/2004) changes:
//   fixed "Click to show" wording for layers
//   added LAYERWS mod
//   fixed a couple IMG tags that were missing width/height attributes
//
```

Finally

Feel free to send comments, suggestions, or bug reports. Officially, this is unsupported code, but unofficially, if there's something wrong with it or there's a useful addition, and I find the time to make the change, then I'll post updates as possible.